WHAT IS CLAIMED IS:

1. A program optimization method for converting program source code written in a programming language into machine language comprising the steps of:

analyzing a target program and detecting exception generative instructions, which may generate an exception, and exception generation detection instructions, which branches a process to an exception process when an exception occurrence condition is detected and an exception has occurred;

dividing said exception generation detection instructions into first instructions, for the detection of exception occurrence conditions, and second instructions, for branching processes to exception processes when said exception occurrence conditions are detected; and

establishing dependencies among program instructions, so that when one of said exception occurrence conditions is detected the process is shifted from a first instruction to a second instruction, and so that when none of said exception occurrence conditions are detected the process is shifted from a first instruction to an exception generative instruction.

2. The program optimization method according to claim 1, following said step for establishing said dependencies among the instructions, further comprising a step of:

collecting multiple second instructions obtained based on multiple exception generation detection instructions detected within a predetermined range of said program. Thus, overall determination is performed based on the detection of said exception occurrence conditions by multiple first instructions, which are obtained based on multiple exception generation detection instructions.

3. The program optimization method according to claim 2, wherein said step of dividing said exception generation detection instructions includes a step of:

generating flag setting instructions, as said first instructions, when said exception occurrence conditions are detected; and wherein said step of collecting said second instructions includes a step of:

detecting the occurrence of exceptions, when flags are set for at least one of said multiple first instructions obtained based on said multiple exception generation detection instructions, and

generating instructions for shifting processes so that said first instructions exchange positions with said second instructions.

4. The program optimization method according to claim 1, following said step of detecting said exception generative instruction, further comprising a step of:

generating compensation code, when an exception generative instruction is accompanied by side effects, for

maintaining order restrictions that are present before said exception generation detection instruction is divided and that concern said side effects.

5. The program optimization method according to claim 1, following said step of setting said dependencies for the instruction, further comprising a step of:

defining said first instruction as a conditional branch and allocating a predicate so as to reflect said conditional branch.

6. The program optimization method according to claim 1, following said step of setting said dependencies for the instructions, further comprising a step of:

defining a first instruction as a conditional branch, and according to the result of code scheduling, generating a compensation code at a branching destination for said first instruction, so as to establish said order restrictions concerning said exception generation detection instruction before said exception generation detection instruction is divided into said first and said second instructions.

7. A compiler for converting the source code for a program written in a programming language into machine language and for optimizing said program comprising:

a graph generator for analyzing a target program and for generating a graph showing the dependencies of operations in said program;

a graph editing unit for editing said graph and for

reducing order restrictions imposed on said operations due to the occurrence of an exception; and

a code reproduction unit for generating program code that reflects said dependencies of said operations of said edited graph;

wherein said graph editing unit detects an exception generative instruction, which may generate an exception, and an exception generation detection instruction, which branches a process to an exception process when an exception occurrence condition is detected and an exception has occurred, divides said detected exception generation detection instruction into a first instruction, which detects said exception occurrence condition, and a second instruction, which branches said process to said exception process when said exception occurrence condition is detected, and establishes a dependency among the instructions, so that the process is shifted from said first instruction in said graph to said second instruction when said exception occurrence condition is detected, or so that the process is shifted from said first instruction to said exception generative instruction when an exception occurrence condition is not detected.

8. The compiler according to claim 7, wherein said graph editing unit removes from said graph order restrictions that are present before said exception generation detection instruction is divided and that concern said exception generative instruction, and order restrictions that are

present before said exception generation detection instruction is divided and that precede said exception generation detection instruction.

9. The compiler according to claim 7, wherein said graph editing unit synthesizes said multiple second instructions obtained based on said multiple exception generation detection instructions in said graph within a predetermined range of said program.

10. The compiler according to claim 7, wherein said graph editing unit sets order restrictions for said exception generative instruction in said graph, so that, when an exception generative instruction is accompanied by side effects, said order restrictions are maintained that are present before said exception generation detection instruction is divided and that concern said side effects.

11. The compiler according to claim 7, wherein said graph editing unit defines a first instruction as a conditional branch, and generates a compensation code at a branching destination for said first instruction, so as to establish said order restrictions concerning said exception generation detection instruction before said exception generation detection instruction is divided into said first and said second instructions.

12. A compiler for converting the source code for a

program written in a programming language into machine language and for optimizing the program comprising:

an intermediate code generator for converting said source code into editable intermediate code;

an optimization unit for optimizing said intermediate code; and

a machine language code generator for using said obtained intermediate code to generate machine language code,

wherein said optimization unit analyzes said intermediate code for said program, transforms said program, within a predetermined range, so that an exception generative instruction that may generate an exception is executed before other instructions, establishes dependencies among the instructions of said program, so that, when an exception has been generated by said exception generative instruction, following the execution of said exception generative instruction the execution of instructions is inhibited, and converts an area in said program, within a predetermined range, so that the occurrence of an exception is detected when an exception is generated by at least one of multiple exception generative instructions, and a process is shifted to a corresponding exception process.

13. The compiler according to claim 12, wherein, when said exception generative instruction is accompanied by side effects, said optimization unit establishes order restrictions for said exception generative instructions in

said graph, so that before said program is transformed the order restrictions concerning said side effects are stored.

14. A computer system comprising:

an input device for receiving a source code of a program;

a compiler for converting said program and converting said program into a machine language code; and

a processor for executing said machine language code of said program,

wherein said compiler analyzes said program, detects an exception generative instruction, which may generate an exception, and an exception generation detection instruction, which branches a process to an exception process when an exception occurrence condition is detected and an exception has occurred, divides said detected exception generation detection instruction into a first instruction, which detects said exception occurrence condition, and a second instruction, which branches said process to said exception process when said exception occurrence condition is detected, and establishes a dependency among the instructions, so that the process is shifted from said first instruction to said second instruction when said exception occurrence condition is detected, or so that the process is shifted from said first instruction to said exception generative instruction when an exception occurrence condition is not detected.

15. The computer system according to claim 14, wherein said compiler collects a plurality of said second instructions obtained by a plurality of said exception generation detection instructions, so as to collectively determine that an exception occurrence condition is detected from a plurality of said first instructions that are obtained based on a plurality of said exception generation detection instructions within a predetermined range of said program.

16. A storage medium on which input means of a computer stores a computer-readable program that permits said computer to perform:

a process for analyzing said program, and detecting an exception generative instruction, which may generate an exception, and an exception generation detection instruction, which branches a process to an exception process when an exception occurrence condition is detected and an exception has occurred;

a process for dividing said detected exception generation detection instruction into a first instruction, which detects said exception occurrence condition, and a second instruction, which branches said process to said exception process when said exception occurrence condition is detected; and

a process for establishing a dependency among the instructions, so that the process is shifted from said first instruction to said second instruction when said exception occurrence condition is detected, or so that the process is shifted from said first instruction to said exception

generative instruction when an exception occurrence condition is not detected.

17. The storage medium according to claim 16, wherein, following said process for setting said dependency among the instructions, said program stored thereon further comprises a process for collecting a plurality of said second instructions obtained by a plurality of said exception generation detection instructions within a predetermined range of said program; and wherein said program collectively determines that an exception occurrence condition is detected from a plurality of said first instructions that are obtained based on a plurality of said exception generation detection instructions.

18. A program transmission apparatus comprising:

storage means for storing a program that permits said computer to perform

a process for analyzing said program, and detecting an exception generative instruction, which may generate an exception, and an exception generation detection instruction, which branches a process to an exception process when an exception occurrence condition is detected and an exception has occurred,

a process for dividing said detected exception generation detection instruction into a first instruction, which detects said exception occurrence condition, and a second instruction, which branches said process to said

exception process when said exception occurrence condition is detected, and

a process for establishing a dependency among the instructions, so that the process is shifted from said first instruction to said second instruction when said exception occurrence condition is detected, or so that the process is shifted from said first instruction to said exception generative instruction when an exception occurrence condition is not detected; and

transmission means for reading said program from said storage means and transmitting said program.

19. The program transmission apparatus according to claim 18, wherein, following said process for setting said dependency among the instructions, said program stored thereon further comprises a process for collecting a plurality of said second instructions obtained by a plurality of said exception generation detection instructions within a predetermined range of said program; and wherein said program collectively determines that an exception occurrence condition is detected from a plurality of said first instructions that are obtained based on a plurality of said exception generation detection instructions.